[illegible]

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/735,513	12/11/2003	Joachim Bender	13913-115001 / 2003P00326	1575

32864 7590 07/02/2007
FISH & RICHARDSON, P.C.
PO BOX 1022
MINNEAPOLIS, MN 55440-1022

EXAMINER	
KHATRI, ANIL	
ART UNIT	PAPER NUMBER
2191	
MAIL DATE	DELIVERY MODE
07/02/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	**Application No.** 10/735,513	**Applicant(s)** BENDER, JOACHIM
	Examiner Anil Khatri	**Art Unit** 2191

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE _3_ MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on _11 December 2003_.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) _1-21_ is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) _1-21_ is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on _11 December 2003_ is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _8/16/04_.	6) ☐ Other: _____.

DETAILED ACTION

Specification

The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed.

The following title is suggested: *"Using Incremental Generation to Develop Software Applications"*.

The use of the trademark UML etc. has been noted in this application. It should be capitalized wherever it appears and be accompanied by the generic terminology.

Although the use of trademarks is permissible in patent applications, the proprietary nature of the marks should be respected and every effort made to prevent their use in any manner which might adversely affect their validity as trademarks.

The disclosure is objected to because it contains an embedded hyperlink (see page 9) and/or other form of browser-executable code. Applicant is required to delete the embedded hyperlink and/or other form of browser-executable code. See MPEP § 608.01.

Claim Objections

Claim 6 is objected to because of the following informalities: use of abbreviation MDO, at least once it should be spelled out. Appropriate correction is required.

Claim Rejections - 35 USC § 101

35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

Claims 1-21 are rejected under 35 USC 101 because they disclose a claimed invention that is an abstract idea as defined in the case *In re Warmerdam*, 33, F 3d 1354, 31 USPQ 2d 1754 (Fed. Cir. 1994).

Analysis: Claims 1-21 disclosed by the applicant as being a "computer program product..". Since the claims are each a series of steps to be performed on a computer the processes must be analyzed to determine whether they are statutory under 35 USC 101.

Examiner interprets that claims 1-21 are non-statutory because claim recites computer program product are program, per se i.e. the description or expressions of the program are not physical things nor are they statutory process as they do not act being performed. Computer programs do not define any structural and functional interrelationship between the computer program and other claimed aspect of the invention which permits the computer program's functionality could be realized. Therefore, computer program is merely a set of instructions capable of being executed by a computer, the computer program itself is not a process. Further, computer program itself is not a process and without the computer-readable storage medium so its functionality cannot be realized for a practical application. Applicant submit no substance that how this will be processed without incorporating a processor, memory and medium.

Further, examiner interprets that claims 1-21 are not limited to tangible embodiments in view of applicant's disclosure, specification page 22 lines 11-21 the medium is not limited to tangible embodiments, instead being defined as including both tangible embodiments (e.g., [computer readable medium]) and intangible embodiments (e.g., [transmission media, radio frequency (RF), infrared (IR), a carrier wave, telephone line, a signal, etc.]). As such, the claim is not limited to statutory subject matter and is therefore non-statutory. To overcome this type of 101 rejection the claims need to be amended to include only the physical computer media and not a transmission media or other intangible or non-functional media. For the specification at the bottom, carrier medium and transmission media would be not statutory but storage media would be statutory.

Claim Rejections - 35 USC § 112

The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

Claim 6 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

Claims 1-21 are rejected under 35 U.S.C. 102(e) as being anticipated by *Schloegel et al* USPN 7,219,328.

Regarding claims 1-5, 7, 9-12 and 17-21

Schloegel et al teaches,

identify a first main development object (column 9, lines 4-18, As another example of stereotype specialization, in order to produce thread specific code using only a UML model, the concept of threads needs to be introduced within the UML model. One way of introducing the concept of threads within the graphical modeling notation is to create a stereotype named "Thread." However, thread entities should not be treated the same as other UML classes during code generation. For example, a class declaration should not be generated for a Thread class within a C++ header file. Using model-based composable code generation, stereotype specialization can be used to ensure that all Thread instances point to a specialized code generator routine that will take precedence over the base code generator routine that typical UML classes share. Hence, instances of Thread objects will be treated appropriately whenever code is generated for the UML model);

identify main development objects related to the first main development objects by an aggregation relationship (column 10, lines 9-14, Each portal on an archetype instance is associated with a corresponding portal in some internal model. In the case of FIGS. 3 and 4, the internal model is the UML model shown in FIG. 3. Arcs on either side of corresponding portals are logically matched. Hence, as shown in FIG. 4, the Line class logically has a realization arc to the Component class);

identify main development objects related to the first main development object by an association relationship (column 10, lines 9-14, Each portal on an archetype instance is associated with a corresponding portal in some internal model. In the case of FIGS. 3 and 4, the internal model is the UML model shown in FIG. 3. Arcs on either side of corresponding portals are logically matched. Hence, as shown in FIG. 4, the Line class logically has a realization arc to the Component class) determine if any identified main development objects have changed (column 8, lines 4-7, Although a specific order of preference has been described above, the order of precedence may be different than that stated above and may change from design to design) ; and

re-generate the first main development object if any identified main development objects have changed (column 9, lines 4-18, s another example of stereotype specialization, in order to produce thread specific code using only a UML model, the concept of threads needs to be introduced within the UML model. One way of introducing the concept of threads within the graphical modeling notation is to create a stereotype named "Thread." However, thread entities should not be treated the same as other UML classes during code generation. For example, a class declaration should not be generated for a Thread class within a C++ header file. Using model-based composable code generation, stereotype specialization can be used to ensure that all Thread instances point to a specialized code generator routine that will take precedence over the base code generator routine that typical UML classes share. Hence, instances of Thread objects will be treated appropriately whenever code is generated for the UML model).

Regarding claim 6

Schloegel et al teaches,

wherein identifying the second main development object comprises identifying a non-MDO development object as a starting point and asking the non-MDO development object what its MDO is (column 7, lines 34-47, in model-based composable code generation, a graphical modeling entity may have several code generator routines attached to it or to its meta-entity or to one or more collections of which it is a member. For example, a graphical modeling entity may have a base code generator routine attached to its meta-entity and one or more specialized code generator routines attached to it. Accordingly, when such a graphical modeling entity is queried for its attached code generator routine, this graphical modeling entity selects from a number of available code generators that can be executed. This set of available code generators is initially limited to those associated (via an index key) with the type of code that is specified. Examples include, but are not limited to, Java source code, C++ source code, documentation, and middleware configuration files. A single routine must be selected from this set to be executed).

Regarding claim 8

Schloegel et al teaches,

Main development objects corresponds to as file borders (column 12, lines 28-36, A key constraint for documentation is that no two entities have the same name. Otherwise, ambiguous documentation will result. However, when multiple archetype instances that share implementations appear together in the same model, the use of the base Name routine will result in duplicate names. FIGS. 4 and 5 give an example. FIG. 5 shows a second use of the Composite Archetype. (The purpose of this sub-model is to be able to treat a number of directories and files as a single logical file system).

Regarding claims 13-15

Schloegel et al teaches,

a generator comprising (figures 1-3, column 3, lines 52-60, In accordance with one aspect of the present invention, a method of generating code in a model-based development environment comprises the following: modeling a system using graphical modeling entities; attaching a modular code generator routine directly to at least one of the graphical modeling entities; traversing through the graphical modeling entities to access the code generator routine; and, executing the accessed code generator routine so as to generate code for the system);

a search module configured identify a first main development object, to identify main development objects related to the first main development objects by an aggregation relationship, and to identify main development objects related to the first main development object by an association relationship (column 4,lines 7-15, In accordance with still another aspect of the present invention, a method of developing code comprises the following: attaching at least one modular code generator routine directly to each of a plurality of graphical modeling entities; traversing through the graphical modeling entities; accessing the code generator routines of each traversed modeling entity;

selecting only one modular code generator routine per graphical modeling entity; and, executing each of the accessed code generator routines so as to generate the code);

a comparator module configured to determine if any identified main development objects have changed (column 6, lines 40-49, A major benefit of model-based composable code generation is that only a single unified compositional hierarchy exists because the code generator routines take on the hierarchy of the graphical modeling notation and/or the instance models. Thus, the problems of the dual hierarchy of the prior art are avoided. Additionally, the close integration between graphical models and code generator routines makes it easier to reason and/or prove qualities about generated code compared to current code generation techniques; and a generation module configured to re-generate the first main development object if any identified development objects have changed (figures 1-3, column 8, lines 27-47, As shown in FIG. 2, base code generator routines 32, 34, and 36 are attached to corresponding graphical meta modeling entities 38, 40, and 42 of the meta model 24. These base code generator routines are default code generator routines that are used in the absence of further specialization code generator routines when code is generated for any modeling entity instance of the type that they define in any instance model including the instance model 20. The base code generator routine 32 for all instances of UML classes is attached to the class meta-entity 38. Similarly, the base code generator routine 34 for all UML methods is attached to the method meta-entity 40, and the base code generator routine 36 for all UML attributes is attached to the
attribute meta-entity 42. It is noted that, for illustrative purposes only, the base code generator routines 32, 34, and 36 (as well as their links to the corresponding graphical modeling entities 38, 40, and 42) are shown visually. However, the base code generator routines 32, 34, and 36

and their links (attachments) to the corresponding graphical modeling entities 38, 40, and 42 need not be visible, except when specific modeling entities are explicitly examined).

Regarding claim 16

Schloegel et al teaches,

The comparator module is further configured to store the identified main development object in the cache (figures 2 and 7, column 6, lines 40-49, A major benefit of model-based composable code generation is that only a single unified compositional hierarchy exists because the code generator routines take on the hierarchy of the graphical modeling notation and/or the instance models. Thus, the problems of the dual hierarchy of the prior art are avoided. Additionally, the close integration between graphical models and code generator routines makes it easier to reason and/or prove qualities about generated code compared to current code generation techniques.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Anil Khatri whose telephone number is 571-272-3725. The examiner can normally be reached on M-F 8:30-5:00 PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Zhen can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

***

[illegible]

ANIL KHATRI
PRIMARY EXAMINER